C. Supporting Research and Technology

1. An Executive Program for Telemetry Simulation,

J. A. Flynn

a. Introduction. In preceding articles, various features of the supervisory program (SPS 37-57, Vol. II, pp. 117–121), the user-oriented source-language (SPS 37-58, Vol. II, pp. 87–97), and means of communication among program components (SPS 37-59, Vol. II, pp. 74–78) have been described. The present article describes the manner in which simulated engineering telemetry is produced.

b. Input. The prototype for these researches in spacecraft simulation is the *Mariner* Mars 1969 vehicle. The simulation described, therefore, is structured about a pseudonoise (PN) sequence to identify position in the telemetry bit stream, a high-rate deck comprising multiple subcommutated channels, and a code word to identify the effects of subcommutation.

The foregoing structure is enabled by user-supplied input of the items described in Table 11. The syntax of these inputs is displayed in Table 12; the notation is

Table 11. Inputs for commutator definition

PNSEQ	An arbitrary sequence of 15 *zeros* and *ones* to define the PN sequence. The Hollerith 0–1 sequence will be mapped one-for-one into the PN bit string. Alternatively, input the letter "D" followed by the appropriate 5 octal digits.
DECK	A sequence of integer pairs naming the first and last channels in each deck.
SUBCOM	A sequence of integer pairs connecting each subcommutated channel with the first channel of the deck it points to.
CHANT	Channels to be computed as explicit functions of time.
INDEX	A sequence of ordered $(n+1)$-tuples, to define the index word associated with each combination of subcommutated channels.

Table 12. Syntax of commutator input[a]

$$
\begin{aligned}
&<\text{commutator input}> \;::= <\ell><\text{com. hdr}><\ell>\{[<p>\,|\,<d>\,|\,<s>\,|\,<c>\,|\,<i>]\}_5^{5/<\ell>}\\
&<\ell> \;::= \text{a new line}\\
&<\text{com. hdr}> \;::= \text{*COMMUTATOR}\,\{\underline{b}\}_{61}^{61}\\
&<p> \;::= \text{PNSEQ} = \{0\,|\,1\}_{15}^{15}\,|\,D\{0\,|\,1\,|\,2\,|\,3\,|\,4\,|\,5\,|\,6\,|\,7\}_5^5\\
&<d> \;::= \text{DECK} = \{<C_F> - <C_L>\}_1^{/'}\\
&<s> \;::= \text{SUBCOM} = \{(<S>, <C_F>)\}_1^{/'}\\
&<c> \;::= \text{CHANT} = \{<C>\,|\,<C> - <C>\}_1^{/'}\\
&<i> \;::= \text{INDEX} = \{(<\sigma_1>, \cdots, <\sigma_n>, <i\text{word}>),\}_1\,(CCS\,\underline{b}\,RO, <i\text{word}>)\\
&<\textit{blank}> \;::= \underline{b}\\
&<C> \;::= \text{a channel name}\\
&<C_F> \;::= \text{name of first channel in deck}\\
&<C_L> \;::= \text{name of last channel in deck}\\
&<S> \;::= \text{name of a subcommutated channel}\\
&<\sigma_k> \;::= \text{name of channel pointed to by } k\text{th subcommutated channel}\\
&<i\text{word}> \;::= \text{decimal digit} \leq 63
\end{aligned}
$$

[a]Unless the contrary is indicated, embedded blanks are ignored.

adapted from Lee (Ref. 1) and is described in *Paragraph e.* Table 13 displays the *Mariner* Mars 1969 commutation inputs. To facilitate both narrative and computational references to the inputs, the definitions in Table 14 have been devised.

In Tables 11–14, the terms "deck name," "channel name," and "channel index number" are used. The "channel name" is the number by which the channel is known, e.g., 100, 213, 437. The "channel index number" is the ordinal number of the channel in question, given that the channels are arranged in channel-name sequence. The "deck name" is the same as the name of the first channel in the deck.

The following relationships are established by inspection.

$$C_{ij} = i + D_j - 1 \qquad (1)$$

$$K_{ij} = i + \sum_{v=0}^{j-1} N_v, \qquad N_0 = 0, j \geq 1 \qquad (2)$$

Eliminating i between Eqs. (1) and (2) and introducing F_j from Table 14, there results

$$K_{ij} = C_{ij} - F_j \qquad (3)$$

where

$$F_j = D_j - \sum_{v=0}^{j-1} N_v - 1 \qquad (4)$$

Table 15 shows the way the *Mariner* Mars 1969 DECK and SUBCOM data are transformed into program-usable form.

c. Environment. The program enters the telemetry publication phase following step termination. During

Table 13. *Mariner* Mars 1969 commutator-defining inputs

```
*COMMUTATOR
PNSEQ   = D03545
DECK    = 100-109, 110-119, 200-209, 210-219, 220-229, 300-309,
          400-409, 410-419, 420-429, 430-439
SUBCOM  = (103,200), (104,210), (110,220), (202,300), (211,400),
          (211,410), (212,420), (212,430)
CHANT   = 305, 401, 404-414, 416-419, 430-439
INDEX   = (200,210,220, 1), (203,213,223, 2), (204,214,224, 3),
          (205,215,225, 4), (206,216,226, 5), (207,217,227, 6),
          (208,218,228, 7), (209,219,229, 8), (201,400,221, 9),
          (201,401,221,10), (201,402,221,11), (201,403,221,12),
          (201,404,221,13), (201,405,221,14), (201,406,221,15),
          (201,407,221,16), (201,408,221,17), (201,409,221,18),
          (300,420,222,19), (301,421,222,20), (302,422,222,21),
          (303,423,222,22), (304,424,222,23), (305,425,222,24),
          (306,426,222,25), (307,427,222,26), (308,428,222,27),
          (309,429,222,28), (201,410,221,41), (201,411,221,42),
          (201,412,221,43), (201,413,221,44), (201,414,221,45),
          (201,415,221,46), (201,416,221,47), (201,417,221,48),
          (201,418,221,49), (201,419,221,50), (300,430,222,51),
          (301,431,222,52), (302,432,222,53), (303,433,222,54),
          (304,434,222,55), (305,435,222,56), (306,436,222,57),
          (307,437,222,58), (308,438,222,59), (309,439,222,60),
          (CCS R/O, 31)
```

that period, all channels which do not depend explicitly on time were evaluated at t_R, the right-hand side of the integration interval, via source language statements of the following sort:

$$C = \mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3 \tag{5}$$

The left side of this equation is an integer denoting the channel name. The $\mathcal{E}$'s are arithmetic expressions, $\mathcal{E}_1$ being the computed telemetry value for the channel, while $\mathcal{E}_2$ and $\mathcal{E}_3$ are factors used to scale $\mathcal{E}_1$. A typical instance of Eq. (5) might be

$$106 = EY, 1.0, 10.0$$

The source statement Eq. (5) gives rise to the target language statement,

$$\text{CALL CHAN}\,(\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, C) \tag{6}$$

Subroutine CHAN performs three functions:

(1) It maps the channel name C into the channel index number K.

(2) It maps $\mathcal{E}_1$ into the interval [0, 1] by

$$\mathcal{E}_1 \leftarrow \frac{\mathcal{E}_1 - \mathcal{E}_2}{\mathcal{E}_3 - \mathcal{E}_2} \tag{7}$$

(3) It stores the scaled $\mathcal{E}_1$ into the Kth cell of the telemetry array ZR.

d. Program action. Given the foregoing array ZR of telemetry values at t_R for all channels (except as noted), the publication problem consists of (1) deciding which channels to publish, (2) determining the publication time for each publishable channel, and (3) obtaining the correct telemetry values for these channels at the proper times. Most of the complexity at this stage can be traced to the requirements for subcommutation.

Table 14. Deck and channel parameters

D_j	name of deck j
N_j	number of channels in deck j
P_j	name of (subcommutated) channel that points to deck j
F_j	quantity that converts deck j channel names into channel index numbers
C	a channel name
K	a channel index number

Subscripts may be used on C or K to draw attention to some specific characteristic:

Subscript	C, K characteristic emphasized
ij	ith channel of deck j
L	first publishable channel in $[t_L, t_R]$
R	last publishable channel in $[t_L, t_R]$
ν	νth C or K in sequence (note that $K_\nu = \nu$ by definition)

Table 15. DECK and SUBCOM inputs (*Mariner* Mars 1969)[a]

DK	Deck number, j									
	1	2	3	4	5	6	7	8	9	10
C_{1j}	100	110	200	210	220	300	400	410	420	430
$C_{N_j j}$	109	119	209	219	229	309	409	419	429	439
P_j	0	0	103	104	110	202	211	211	212	212
N_j	10	10	10	10	10	10	10	10	10	10
F_j	99	99	179	179	179	249	339	339	339	339
δC_{ij}	0	0	0	0	0	0	0	0	0	0

[a]When (1) $P_j = P_{j+1}$ and (2) $C_{1,\,j+1}$ is consecutive with $C_{N_j j}$, columns j and $j+1$ may be collapsed into a single column. This condition occurs for $j = 1$, 7, and 9.

It was shown in SPS 37-57 that publication of the νth high-deck channel, say K_ν, will begin at the following times:

$$t_P(K_\nu) = t_M + (K_\nu - 1)\,T_c - mT_F, \qquad \begin{cases} m = 0, 1, \cdots \\ K_\nu \leq NH \end{cases} \tag{8}$$

Moreover, the publication time of the first high-deck channel in the integration interval $[t_L, t_R]$, say K_L, is related to t_L by

$$t_P(K_L - 1) \leq t_L < t_P(K_L) \tag{9}$$

Taking $\nu = L$ and substituting Eq. (8) into Eq. (9), there results the index number of the earliest high-deck channel to be published, relative to $[t_L, t_R]$:

$$K_L = 2 + \left\lfloor \frac{t_L - (t_M + m_L T_F)}{T_c} \right\rfloor (\mathrm{mod}\, NH) \tag{10}$$

where the auxiliary m_L is given by

$$m_L = \left\lfloor \frac{t_L - t_M}{T_F} \right\rfloor \tag{11}$$

The index number of the last high-deck channel to be published, relative to $[t_L, t_R]$, is then given by

$$K_R = K_L + \left\lfloor \frac{t_R - t_P(K_L)}{T_c} \right\rfloor (\mathrm{mod}\, NH) \tag{12}$$

Corresponding, therefore, to $[t_L, t_R]$, the index numbers of the high-deck channels to be published are those indexes K_ν such that

$$K_L \leq K_\nu \leq K_R \tag{13}$$

Recognizing that in the instance of a subcommutated channel, the high-deck channel is merely an alias of the true channel, it is the responsibility of the locator algorithm to thread through any depth of subcommutation to find the actual channel to be published. A description of this algorithm refers to the array *DK* of Table 15, which is shown as it appears just after reading the DECK and SUBCOM inputs, i.e., in terms of channel names. (See also Tables 13 and 14.) Internally, channel names are inconvenient to use because they do not form a consecutive sequence. For this reason, the program logic is carried out in terms of channel index numbers. Before the information in *DK* is useful, therefore, the channel names C_{1j}, $C_{N_j j}$, and P_j must be transformed by Eq. (3) into channel index numbers, K_{1j}, $K_{N_j j}$, and P_j^K. Subsequent references to *DK* or Table 15 assume this transformation.

Row 6 of Table 15 contains the quantity δC_{ij}, which, for the channel named C_{ij}, is defined by

$$\delta C_{ij} = C_{ij} - C_{1j} \tag{14}$$

Converting names to index numbers by Eq. (3), it is apparent also that

$$\delta C_{ij} = K_{ij} - K_{1j} \tag{15}$$

These equations are useful in the subcommutation process.

The locator algorithm may now be given. The notation employed is essentially that developed in Ref. 2:

(1) $\alpha \leftarrow K_\nu$

(2) $(\exists j \ni \alpha = P_j^K)$; $(T,F) \rightarrow (3,5)$

(3) $\alpha \leftarrow K_{1j} + \delta C_{ij}$

(4) $\delta C_{ij} \leftarrow \delta C_{ij} + 1\,(\mathrm{mod}\, N_j)$; $\rightarrow 2$

(5) $K_{ij} \leftarrow \alpha$; $\rightarrow$ return

Fundamentally, the algorithm states that, if K_ν is *not* a subcommutated channel, then K_ν is the desired index. This is the *false* exit of step 2. If it *is* subcommutated (the *true* exit of step 2), obtain the index of the channel pointed to (step 3: Eq. 15). Since *that* channel may be subcommutated also, the process is iterated until no further subcommutation is detected.

Once the true channel index has been found (i.e., K_ν has been mapped into K_{ij}), the telemetry value for that channel at $t_P(K_\nu)$ can be obtained. The mode of determination depends on whether the simulation function for the channel in question depends explicitly on time. If so, its name will be among the CHANT inputs. The only channels of this class encountered thus far belong to the

temperature subsystem. For such a channel, the supervisor executes the following code:

$$\text{IE} \leftarrow C_{ij}$$

$$\text{KSECT} = 3$$

$$\text{KSUB} = 1$$

$$\text{KCOM} = 60$$

$$\text{GO TO } 3000$$

The result will be the telemetry value of channel C_{ij} at time $t_P(K_v)$, stored in $ZL(K_{ij})$.

If the channel is not found in the CHANT set, its value is obtained by interpolation: $ZL(K_{ij})$ and $ZR(K_{ij})$ contain the telemetry values of C_{ij} at t_L and t_R, respectively; hence,

$$ZL(K_{ij}) \leftarrow \left[\frac{ZR(K_{ij}) - ZL(K_{ij})}{t_R - t_L}\right][t_P(K_v) - t_L] \qquad (16)$$

If there is room in the telemetry output queue, the channel value is placed there to await bit-wise publication under control of the interval timer. If the queue is momentarily full, the supervisor attends to miscellaneous tasks and input/output interrupts that have been accumulated. A subsequent article will investigate the degradation in telemetry publication that may occur in certain critical situations.

e. Language descriptors. The symbols of the metalanguage used to describe the input syntax are defined as follows [after Lee (Ref. 1)]:

$<x>$ read, "the object named x."

$:=$ read, "is formed from."

$|$ exclusive *or*.

$\{z\}_i^{j/\sigma}$ z is to be repeated *in situ* at least i times and not more than j times. Consecutive instances of z are separated by the contents of the field "σ." If i is omitted, its value is assumed to be 1. If j is omitted, its value is assumed to be infinity. If "$/\sigma$" is omitted, σ is assumed to be the null character.

$[R]$ R is a reducing set, defined as follows: Let N denote the full set of elements under consideration. Let A denote the set of elements of N already used, and, therefore, no longer available for use. Let R denote the set of elements of N not yet used, and, therefore, available for use. Then $R = N - A$.

To illustrate the repetition operator, consider $\{A \mid B\}_0^2$. Here, $i = 0$, $j = 2$, $\sigma =$ null. Any of the following constructs is a legitimate production:

null	A	AA
	B	AB
		BA
		BB

By way of contrast, consider also $\{A \mid B\}_0^{2/'}$. In this instance, $i = 0$, $j = 2$, and σ is the symbol ",". Any of the following constructs is legitimate

null	A	A, B
	B	A, A
		B, A
		B, B

To illustrate the reducing set, consider the options field of the IBJOB Fortran compiler control card (Ref. 3) (i.e., the $IBFTC card). This field begins in column 16. It may be null, or it may contain up to five subfields in any order, without repetition, separated by commas. The available options are suitably abbreviated as follows:

Symbol	Option
$<L>$	List
$<D>$	Debug
$<P>$	Punch
$<I>$	Instruction set
$<R>$	Index register

The syntax of the options field may, therefore, be written as

$$<\text{options field}> := \{[<L> \mid <D> \mid <P> \mid <I> \mid <R>]\}_{0}^{5/}$$

References

1. Lee, J. A. N., *The Anatomy of a Compiler.* Reinhold Book Corp., New York, N. Y., 1967.
2. Iverson, K. E., *A Programming Language.* John Wiley & Sons, Inc., New York, N. Y., 1962.
3. *IBM 7090/7094 IBSYS Operating System, Version 13, FORTRAN IV Language,* Form C 28-6390. IBM Systems Reference Library.

2. Modifications to the Syntax of TMG for Purposes of Spacecraft Telemetry Simulation, *R. I. Ścibor-Marchocki*

a. Introduction. The first three articles in this series on spacecraft telemetry simulation discussed certain aspects of the supervisory program (SPS 37-57, Vol. II, pp. 117–121), the user-oriented source language (SPS 37-58, Vol. II, pp. 87–97), and the lines of communication created among various program components (SPS 37-59, Vol. II, pp. 74–78). These articles mentioned that the source code would be translated by a special-purpose compiler.

b. Spacecraft simulation. The transmogrification (TMG) compiler is employed to compile the simulation compiler which translates from the user-oriented spacecraft-simulation language to Fortran IV. The syntax portion of TMG, known as TMGL, has been modified to improve the performance of both the TMGL compiler itself and, indirectly, the simulation compiler.

Of the various compilers, for compiling a compiler, which have been reported (Ref. 1), the TMG compiler, invented and coded by Robert McClure (Ref. 2), possesses these advantages: (1) it is versatile, (2) its parser is capable of backing up, (3) its syntax TMGL is written in its own language and thus can grow easily by a recompilation performed by its previous generation, and (4) the TMG compiler exists, works, and is available (SPS 37-44, Vol. IV, pp. 17–22).

Like most other compiler-compilers, TMG had its problems—incomplete documentation, inadequate diagnostics, lack of or incorrect error recovery, and erratic behavior. As a result, it was difficult to write the source for a compiler. A run frequently terminated unsuccessfully in a meaningless dump or a report by IBLDR of the presence of obscure undefined virtuals. The compilation often yielded a wrong translation. Checkouts were inconclusive because subsequent runs again were subject to the foregoing problems; i.e., the target compiler was unreliable.

The motivation for the work reported herein has been to help TMG realize its potential by improving TMGL—optimization of the source coding and correction of coding errors has benefited the *first* generation of TMGL itself. Optimization of the target coding, expansion of diagnostics, introduction of error recovery, improvement of existing features, introduction of new features, and the addition to the documentation benefited the *second* generation of TMGL itself.

c. TMG compiler. The TMG compiler employs a top-down left-to-right parser with backup. A typical colon-type sentence

$$\text{A . . B/C} \quad \text{D} = \text{E}$$

consists of a label A followed by a colon, zero or more components with optional alternates, and an optional subject E preceded by an equal sign. The left part of a component may be simple or compound, e. g.,

B1'B2'B3

The alternate of a component may be either a name of a colon-type sentence exclusive or a similar sentence, without a label and colon, but enclosed in parentheses, e. g.,

(C1 C2 = C3)

A component that does not possess an alternate may be enclosed in parentheses, and if it is either a name of a character-class exclusive or is enclosed in parentheses, then it may be followed by a star, e. g.,

(D1 D2)

(D)*

D*

The subject may be a name of a definition or a definition, separated by another equal sign if both a name and a definition are present. The subject, at its end, may possess an alternate consisting of the name of a colon-type sentence.

The TMGL compiler parses the source code and translates it into a suitable sequence of assembly language cards. A listing of this intermediate assembly language code is instructive both as an educational aid and as a check upon the compiler coding under whose guidance the translation was accomplished.

*d. *** Functions.* Each component which does not possess parentheses or the BNF| compiles into one computer word consisting of: an operation code, an address indicating the name of the left part, and the decrement either indicating the name of the alternate by an address exclusive or the absence of an alternate by a value of zero. For example,

LP/IMAC LS

compiles into

```
C       LP,,IMAC
C       LS,,0
```

where, for easier readability, we write the assembly language target code, substituting the real names whenever they are available.

The left part of the component may be a function without arguments. Such a function is called a single-star function because it is declared by an equal-sign-type sentence like

TYPVAR = *TYPVAR

A component consisting of this function

TYPVAR

compiles into

```
M       TYPVAR,,0
```

A function with an argument

CHKFLG = **CHKFLG

presents a problem of where to indicate the address of the argument. The solution adopted for such double-star functions is to place the address of the argument in the decrement and to omit any indication of the alternate. The alternate is placed into the preceding word which is inserted and possesses the function ALEX as its left part. For example,

CHKFLG(DEF-FL)/(FV1 RSETKY(FAIL) = R2)

compiles into

```
F4960X C    FV1,,0
       MS   RSETKY,,FAIL
       S    R2,,0
       M    ALEX,,F4960X
       MS   CHKFLG,,DEF-FL
```

The function ALEX is a complicated time-consuming function but essential in the execution of a two-star function.

Certain functions with an argument can never fail because they just perform a specific arithmetical, logical, or housekeeping operation which is defined for all values of the argument. For such a function, the use of ALEX is wasteful of execution storage and especially time, but is not wrong. Such a function now is indicated by a triple star, e. g.,

RSETKY = ***RSETKY

and compiled without the use of ALEX, as already illustrated in the foregoing example.

e. Dictionary. All of the coding involved has been rewritten to increase the use of the dictionary and to provide error recovery and better diagnostics. Extensive use is made of the dictionary to obviate the necessity of detailed parsing and subsequent translation of recurrences of a common expression. The argument of each of the functions

COMPUTE

IF

NOT

is parsed quickly by the sentence BALSTG which looks for strings with balanced, i.e., paired, parentheses. Literal strings are parsed quickly by the sentence DDNDS which looks for strings with balanced, i.e., paired, dollar signs. Such a quick parse hopefully delimits a given entity, which then is entered into the dictionary and checked to see whether it has been parsed in detail and

translated previously. For the same reasons, the definitions now are entered into the dictionary.

A named literal or definition is not entered into the dictionary because a new copy of the coding of the literal or definition is required each time that it is named differently. Therefore, one should make certain that a given literal *or* definition is designated by a unique name, which may be itself.

A set of flags is associated with each word entered into the dictionary. A flag is a binary variable, initialized at false 0. Each time a word is parsed, its flags are interrogated to ascertain compatibility with prior use of the word or are set to indicate present use. Since the compiler performs most of its work on the second pass, each sentence which has an equal sign = rather than a colon .. immediately following the name of the sentence now has to be placed ahead of where that name is used. The only exception is that a definition may be placed either in conformity with the foregoing exclusive or anywhere in the .DEFINITION. section. Suitable diagnostics now are issued if a name is used without being designated, is misused, or is used in non-compatible ways.

Since, at present, the definition is not parsed by TMGL, the semantics of the definition are not checked. For example, no check is made to ascertain whether the argument(s) of a $Pn or $Fn function have been designated already. If anything except an integer constant or a name of a colon-type sentence is used as such an argument without prior designation or without a definition in the .DEFINITION. section, then the *real* name of that argument will become an undefined virtual symbol during the subsequent IBLDR pass.

In the past, if a name were not designated at all or early enough, diagnostics would be unlikely, but, at best, only an obscure alias of the name would appear as an undefined virtual during the subsequent IBLDR pass. At worst, a meaningless dump would result during an execution which invoked that alias.

f. Alternate. Each component may possess an alternate, to be attempted upon the failure of either the next component(s) or the left part of the given component. The alternate has been improved by the introduction of the diagnostics, the error recovery, and the following documentation.

** + I.* The alternate of a component may be indicated in the style of the assembler as * + I, where the star means here and the integer constant indicates how many words hence. This construction is especially popular to indicate an optional item. For example, the sentence

```
OPTC.. $,$/* + 1 = RO
```

compiles into

```
      ZZ101 OCT     730000000000
  OPTCT RACE        584,,22360
          X         ZZ101,,* + 1
          S         = 0,,0
```

To remind the programmer that a double-star function compiles into two words and thus the component count does not equal the word count, four warning messages are available now. For example, the sentence

```
X.. COMPUTE(J1 = 0) IF(I-KEY)/* + 2
    COMPUTE (J1 = 1)
    CVTD(J1) COMPUTE(J1 = 0)
    CHKFLG(ARRAY)/* + 3
    COMPUTE(J1 = 1) CVTD(J1) = Y
```

compiles into

```
(1211X NULL
          CLA       = 1
          STO       J1
          TRA       ALEX
(0211X TSX          PACK,1
          CLA       I-KEY
          TZE       NOGO
          TRA       EXIT
(9111X NULL
          CLA       = 0
          STO       J1
          TRA       ALEX
```

```
X     TRACE     463,,22360
            M   (9111X,,0
            M   (0211X,,* + 2
            M   (1211X,,0
           MS   CVTD,,J1
            M   (9111X,,0
            M   ALEX,,* + 3
           MS   CHKFLG,,ARRAY
            M   (1211X,,0
           MS   CVTD,,J1
            S   Y,,0
```

and causes the warning message

*****THIS * + I MAY NOT WORK

to be written with a pointer to the * + 3. The * − I is used too rarely to justify any checking to be performed by the compiler.

SCAN. The SCAN parser, which is employed by default at each level below the top level, is a left-to-right parser with backup. This backup capability is both useful and dangerous. Consider the fragment of a colon-type sentence

A/C B

If the left part A of the first component fails, then control passes to its alternate C. If A succeeds, control passes to the second component B. Thereupon, if the component B fails, then control reverts, i.e., backs up, to C, the alternate of the successful A. For example, the construction

A/* + 1 B

makes A optional but usually will waste execution time whenever B fails, after a successful A, because then control will back up to the alternate * + I which redirects control to B. Thereupon, B will fail again every time unless the intersection of A and B is non-null. The more complicated construction

(A) B

makes A optional and not subject to backup from B because A is placed one level down. The related construction

(A)* B

makes one or more occurrences of A optional and for the same reason is not subject to backup from B. In each of these last two cases, output, perhaps null, will be produced by the parenthesized expression. This output will correspond to the last output-producing component of A, if any.

Another example, the construction

A/* + 2 B C

will cause control to skip past B directly to C if A fails, but control also will pass to C if B fails after a successful A. The more complicated construction

D/(B) C

where D is the complement of A, solves the problem but creates a new one if C is subject to failure. This new problem is apparent in the resulting target code:

```
D1X C     B,,0
      C   0,,B2X
      C   D,,D1X
B2X NULL
      C   C,,0
```

which is equivalent to the source

D/(B GO-TO-C) C

except that the syntax has no provision for the component go-to-C.

Because the .OR. compiles into a construction involving alternates, it has a backup problem. For example, the construction

A .OR. B/C D

compiles into

```
C   A,,A1X
C   0,,A2X
```

```
A1X NULL
        C       B,,C
A2X NULL
        C       D,,0
```

which is equivalent to the source

```
A/* + 2 **/* + 2 B/C D
```

If A succeeds and then D fails, control reverts to B *via* the alternate of A.

g. Termination. A colon-type sentence may be either terminated by an equal sign = followed by a subject exclusive or left unterminated but followed by another colon-type sentence.

SUBJECT. The SUBJECT has been rewritten completely to treat each of the eight syntactical situations exemplified by

```
P1
$($P1/PZE/0//$)
ARDEF = (4)$($P1(ARBNOP)/S/XP21,,$Q4//
  $Q1/C/$Q3,,O//$)
ARDEF
P1/TMA
= $($P1/CHS//$)/TMA
R321 = $($P3$P2$P1$)/AP1
R321/AS1
```

in a systematic manner and to improve the readability of the source code. As a result, each of the eight choices now is permitted and in each case the same, more extensive, diagnostics are available.

External bypass. For the occasional colon-type sentence that does not end with a subject, the following sentence, which then must be of colon-type, has to provide a bypass around its subsidiary text to cause control to transfer to its beginning from the end of the previous sentence. For example, the typical left-recursive construction consisting of a pair of sentences

```
LS1.. LP/IMAC
LS2.. $V$/( = R1) LP/IMAC = $($P2+$P1)/LS2
```

compiles into

```
LS1 TRACE       173,,22360
        C       LP,,IMAC
        C       ,,L4360X
(3360X OCT      007502002000
        OCT     750100000000
L5360X S        =O007501000000,,0
LS2 TRACE       174,,22360
L4360X X        V$,,L5360X
        C       LP,,IMAC
        S       (3360X,,LS2
```

Now an average of almost a word of storage per colon-type sentence is saved by the avoidance of the unnecessary use of this bypass, consisting of

```
        C       ,,created-name
```

and the same created-name on the card immediately following the starting card

```
label   TRACE   line-number,,22360
```

If a colon-type sentence that does not end with a subject is not followed by another colon-type sentence, an error message is written now.

h. Recovery from warning errors. A colon-type sentence whose name has been employed as an alternate imposes a different requirement upon its subject than does a colon-type sentence whose name has been employed as a left part of a component; therefore, seldom is it possible to use the name of the same colon-type sentence in both syntactical positions. A warning message is printed when such a second use is encountered by the compiler. A similar warning message results if a name of a definition is present as the left part or alternate of a component or as the alternate of a subject. The compilation now is performed suitably for the actual entity which is present. An intentional example of the recoverable errors has been coded:

```
NM8.. $,$/(NM9/R1 R2C1/REQN) NM6/IMAC
  R2C1/RFUN
```

prints the error messages

```
*****ALTERNATE FIELD NAME HAS BEEN
     USED AS A DEF NAME
*****R1
*****NAME USED AS A COMP HAS ALSO
     BEEN USED AS A DEFINITION
*****R2C1
*****ALTERNATE FIELD NAME HAS BEEN
     USED AS A DEF NAME
*****REQN
*****NAME USED AS A COMP HAS ALSO
     BEEN USED AS A DEFINITION
*****R2C1
*****ALTERNATE FIELD NAME HAS BEEN
     USED AS A DEF NAME
*****RFUN
```

with pointers to the items involved and compiles into

```
          C      ,,R4580X
    R5580X S     RFUN,,0
      ZZ45 OCT   730000000000
    R8580X S     REQN,,0
    R9580X S     R1,,0
    R6580X C     NM9,,R9580X
           S     R2C1,,R8580X
           C     0,,R7580X
       NM8 TRACE 318,,22360
    R4580X X     ZZ45,,R6580X
    R7580X NULL
           C     NM6,,IMAC
           S     R2C1,,R5580X
```

i. Macros. Since the TMGL compiler yields an assembly target code, the TMGL compiler has at its disposal all of the features of the following IBMAP assembler, e. g., macro instructions. Each name stored in the dictionary has associated with it a set of binary variables, called flags. The flags are designated by the compiler using the MFLG macro, which now permits up to 18 inclusive flags to be designated. This macro is invoked by, e. g., the source code

```
.FLAGS. ARRAY,DEF-FL,G-FLAG,L-FLAG
```

which compiles into

```
MFLG    ARRAY
MFLG    DEF-FL
MFLG    G-FLAG
MFLG    L-FLAG
```

A new macro has been introduced to permit the declaration of arrays for use by the compilers. This macro is invoked by, e. g., the source code

```
.PTRS. PTR1(10),PTRA1(50),PTRB1(10)
```

which compiles into

```
PRTS    PTR1,10
PRTS    PTRA1,50
PRTS    PTRB1,10
```

The value of the name is the location of the first word of the array, while the size of the array is enclosed by the pair of parentheses. The elements of the array are indirectly addressed, using any variable whose value is the address of the desired element. For example, the component

```
COMPUTE(J2 = PTRA1,(J2) = EQUADR)
```

compiles into

```
ZZ85 NULL
       CLA   PTRA1
       STO   J2
       CLA   EQUADR
       STO*  J2
       TRA   ALEX
```

and stores the name just parsed into the first element of the array PTRA1. The apparently redundant pair of parentheses around J2 compiles as the star—indicating indirect addressing to the subsequent assembler.

j. Linkage. The necessary communication among routines is accomplished *via* the linkage from a virtual name to an entry name.

Entry. Specific allowance now is made for naming any label, i. e., the name of a colon-type sentence or of a definition, as an entry. Each syntax has to have a main-link entry, called PROGRM, which points to the top-level sentence. For example,

```
PROGRAM = PROGRM
PROGRAM.. FORTIO INITIALIZE HEADER
PR1.. TRAILER/PR2 TERMINATE
  DICT/* + 1 **
PR2.. CA EOLMRK/(GLOT EOLMRK **/PR1)
  **/PR1
```

compiles into

```
PROGRM ENTRY      P1840X
P1840X TRACE      115,,22360
             C    FORTIO,,0
             C    INITIALIZE,,0
           etc.
```

Since the PARSDO parser, which automatically is employed at the top level, by itself is a left-to-right parser *without* backup, the top-level sentences have to be written as a loop with a suitable entry and exit(s). Each exit terminates at the null component **. If there is more than one entry pointing to the same label, then the entry (if any) which is named the same as the label must be indicated last.

Virtual. A call to an overlay *via* the virtual OV now could be coded as

```
OVER = *OV..
A.. PARSDO(OVER) = NULL
```

which would compile into

```
A TRACE      211,22360X
        M    ALEX,,0
       MS    PARSDO,,OV
        S    = 0,,0
```

The corresponding entry in the overlay could be coded as

```
PROGRAM = OV
PROGRAM.. A
B.. C D/B **
```

which would compile into

```
OV ENTRY      P1200X
P1200X TRACE  17,,22360
           C  A
           C  ,,B2700X
   B TRACE    18,,22360
  B2700X C    C,,0
           C  D,,B
           C  0,,0
```

A virtual name F of a definition E now is indicated by a trailing equal sign, e. g.,

```
E = *F =
```

When TMG operates in its two-pass mode, TMG cannot compile more than one deck. Since TMGL employs the two-pass mode, TMGL cannot compile more than one deck at a time. Therefore, neither the initial compilation nor any subsequent recompilation of more than one segment of an overlay compiler can be stacked with the execution of that compiler in a single computer run.

k. New features. The new features introduced into TMGL are summarized here for the convenience of a reader who already is familiar with TMG.[3]

Two options have been added:

```
NOT-size
PREDEFined
```

Each of the size options now defaults to the value 25. Each size indicates an upper bound on the length of the

[3]Germann, D. A., *TMG, A Syntax-directed Compiler*, Sep. 23, 1967 (JPL internal document).

named expression which will be entered into the dictionary in the hope of eliminating a duplication(s). The names

R0 P0 NULL

R1 P1

for the definitions

$(\$)

$(\$P1\$)

respectively, are known in most places as PREDEFined if the option is selected. The use of this option saves compilation time and dictionary space. Since there are a few places, e. g., error recovery and arguments of the $Pn function, where these names are unknown as PREDEFined, whichever of these names are used should be defined, e. g., in the .DEFINITION. section.

The triple star has been introduced as another *explicit* type for unknown functions. Thus, nothing was lost by dropping each of the triple-star functions from the known status.

A virtual name of a colon-type sentence or of a definition now may be designated by, e. g.,

A1 = *V1..

A2 = *V2 =

respectively.

A double-word octal constant now may be designated by, e. g.,

A = 123456,654321B

where the B indicates that the preceding one or two words are to be read by TMGL in "binary," i. e., octal. Such a double-word "character class" is required as the argument of each of the functions

**CHAR

***STRING

The macro .PTRS. now is part of the syntax. This macro is used to define arrays, e. g.,

.PTRS. PTR1(10),PTRA1(50)

Since the only functions which now remain known for use as a left-side of a component are

IF

COMPUTE

ARBNO

NOT

any other function has to be declared before it is employed as a left-side of a component. All other names now are available for free use by the programmer.

A definition now may be placed anywhere in the .SYNTAX. section, but ahead of its use as a left-part of a subject. The two other locations still are permissible for the definition.

The apostrophe ' now is available as a synonym for .OR. in a compound left-part of a component. For example, the construction

A'B/C

is much clearer than

A .OR. B/C

but yields the same target code.

A colon-type sentence, without the label and colon, now may be the argument of a double- or triple-star function. This new feature is required for certain uses of the

**LOCAL

**PARSDO

functions. For example,

```
COMPONENT-LIST.. LOCAL(SPI CLO SPI4          $
  = (1)$($P2(BV1)$)) = (1)$($P1(BV1)$)
```

compiles into

```
X2X OCT     000001007501
    OCT     010000000000
```

```
        PTR     BV1
        OCT     000000000000
X4X     OCT     000001007502
        OCT     010000000000
        PTR     BV1
        OCT     000000000000
X1X     C       SPI,,0
        C       CLO,,0
        C       SPI4,,0
        S       X4X,,0
CLST    TRACE   267,,22360
        M       ALEX,,0
        MS      LOCAL,,X1X
        S       X2X,,0
```

The default subject

$= \$(\$P1\$)$

would be unable to pass down the bound variable BV1.

The second arithmetic expression of the arithmetical relation now defaults to zero, e. g.

A.LT.

is a short form for

A.LT.0

l. Conclusion. The guiding principle has been to make both TMGL and the syntax as systematic as possible. That principle, together with attention to detail throughout, has resulted in greater compactness of the source code and in improved readability, reliability, diagnostics, and error recovery. In spite of the greater sophistication, a reduction in source and target coding of TMGL itself and of a typical compiler compiled by TMGL has been effected. Furthermore, any target coding not generated need not be assembled, stored, or executed afterwards; hence, a substantial savings in assembly, execution, and compile time results. The *second* generation of any of the few compilers that, like TMGL, is written in its own language benefits from its own improvements.

As a result, any compiler, e. g., TMGL or the simulation compiler, became easier to write and more likely to compile successfully, compiled and executed faster, used less storage for itself hence could translate larger programs because more space was available for the dictionary, and provided more reliable performance.

References

1. Feldman, J. and Gries, D., "Translator Writing Systems," *Communications of the ACM*, Vol. 11, No. 2, pp. 77–113, Feb. 1968.
2. McClure, R. M., "TMG—A Syntax Directed Compiler," in *Proceedings of the 20th National Conference,* Cleveland, Ohio, August 24–26, 1965. Association for Computing Machinery, 1st edition, ACM Publication P-65, pp. 262–274. Lewis Winner, New York, N.Y., Aug. 1965.

3. A Formalism for Telemetry Decommutation, *J. Kulick*

a. Introduction. Telemetry decommutation is basically a problem in pattern recognition. Because of this, the various classical approaches to pattern recognition problems should be applied to this specific one. Two primary approaches to pattern recognition are statistical decision theory techniques (statistical pattern recognition) and linguistic techniques (linguistic pattern recognition). Statistical techniques are used primarily in cases where there is a classification task. That is, an unknown stimulus is to be classified into one of a pre-defined number of classes. Statistical techniques are also used where a great deal is known about the statistical structure to be encountered. Linguistic techniques are used primarily when the recognition task involves detecting structural relationships among entities. The primary goal here is to parse a given surface structure into its constituent elements and the relationships between these elements.

Telemetry decommutation represents a particularly interesting pattern recognition problem since aspects of both linguistic and statistical processing are inherent in it. In this article, both approaches are used in an integrated system. The linguistic part of the system generates alternative parses of the input telemetry string, and associates, with each alternative parse, values representing the difficulty of the parse, the "difficulty" being measured by heuristic techniques. Each of the alternative parses are then evaluated, using heuristic and statistical information to select the best one. Heuristic and statistical information is also used during the parsing process to reject uninteresting parses.

The approach taken here is based on previous work by Duda and Hart (Ref. 1). Their problem of recognizing handwritten Fortran is similar in structure to the telemetry decommutation problem.

b. The problem. The telemetry decommutation problem is basically one of taking a string of 1's and 0's, and reconstructing the original values that were associated with its construction. Telemetry streams are normally generated by taking the output of a shift register and transmitting it bit by bit. This shift register, or commutator, is broken up into a number of channels. Each channel is assigned a fixed number of bits. Each channel either represents a directly sampled measurement, such as a physical system state, or is subcommutated into a number of subchannels. The subchannels are, of course, sampled at a lower rate. Figure 26 shows a simple commutation with one subchannel. The basic problem, that of deconcatenating or decommutating the channels from the telemetry stream is complicated by the introduction of errors. Errors are introduced in the communication network. The goal now becomes one not only of deconcatenating the telemetry stream, but of choosing, from among the possible deconcatenations, the best one.

The problem can be thought of in linguistic terms. The data transmitted from each channel is considered to be a letter in an alphabet. A *word* in the language consists of the output of one pass through the high deck of the commutator. For example, in Fig. 26 a word would be either B1 N2 B2 H2 or B1 N2 Clock H2. The structure of each word is given by rules of grammar defined by the commutator structure. To *parse* the data is meant to identify the letters (or *terminals*) and deduce the value of identified terminals for each valid (conforming to the rules of grammar) word.

There are essentially two types of parse, top down and bottom up. The top-down parse assumes a structure, and searches for supporting terminals, while the bottom-up parse looks for terminals, and deduces the structure.

We are approaching the telemetry decommutation problem from a bottom-up parse point of view. Therefore, the terminals must be identifiable. We now come to the first statistical aspect of the problem. The various bits associated with a channel are identifiable only with some statistical reliability that they are indeed the correct bits for the channel.

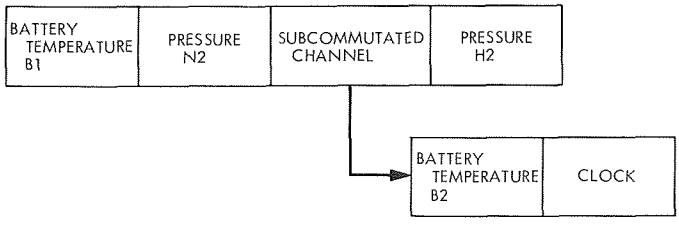

Fig. 26. A simple commutator

The statistical characteristic is caused by two distinct types of phenomena. Firstly, the physical phenomena being observed have probability distributions. For example, if the expected value of a battery temperature is 72°C, there is a non-zero probability that a measurement will yield 71, 73, 70, or 74°C. These distributions are caused by the physical process being observed. So, if we had located what we believed to be the correct set of bits for a particular channel, we could only say that with some probability that these were indeed the correct bits. Secondly, errors with known probabilities are introduced by the communications channel, e.g., bits are received incorrectly, bits are added or dropped, etc. During the parse, occurrences of these errors are hypothesized depending on the probabilities.

The parse will be accomplished by means of a heuristic tree search. The problem is represented by a decision tree where each node or position in the tree represents a particular step or subgoal in the process. Each arc represents a rule of the system that enables one to proceed from one subgoal to the next. There is an evaluation function that assigns a value to each node. The object of the search is to proceed from the root of the tree to the highest level of the tree (representing a complete parse), ending on the node with a maximum value.

The most difficult part of any heuristic system is to develop the evaluation functions to be used. If we were to pursue all nodes with all possible rules, we would have an inordinate number of nodes. What we want here is a way to tell us just how good a position is, and whether or not it is worthwhile pursuing in comparison with the other positions available. The goal of the evaluation function is to eliminate as many nodes as possible.

The specific features of a position that are used in the system will be discussed more fully in *Paragraph c.* In the proposed system, there will be two types of evaluation functions. One will be a position evaluation function, the other a rule evaluation function. The position evaluation will determine what positions to use as the next starting point, while the rule evaluation function will tell what rule to use once a starting position is chosen.

c. Formalization of the problem. In the heuristic tree-search formulation of the problem, we have positions or nodes representing alternative partial interpretations of the bit strings, and a set of rules such that when a rule

is applied to a position, a new position or positions are obtained.

Π = a set of all positions

$\mathbb{R}$ = a set of all rules

π = a single position ϵ Π

R = a single rule ϵ $\mathbb{R}$

We must be able to calculate for every position π a value that tells us, relative to the other positions, how good a position it is:

$$V_{\Pi}(\pi) = \text{value of position } \pi$$

At the highest level, each rule is talking about one channel, and for each channel, there is one and only one rule. To determine whether a rule is applicable to a position, we have a rule–position evaluation function.

$V_{\mathbb{R}}(R,\pi)$ = value of rule R with respect to the position π

Finally, again at the highest level, we have a transform map which applies a rule to a position. In general, a rule applied to a position yields many new positions, each with a different value, each due to a different hypothesis within the rule.

$$T_{PR} : \mathbb{R} \times \Pi \rightarrow \text{power set } (\Pi)$$

We have a transform axiom that tells us just when this transformation map is defined

$$\begin{aligned}(\forall \pi)\,(\forall R)\,[&(R \epsilon \mathbb{R}) \text{ and } (\pi \epsilon \Pi)\\ &\text{and } (V_{\Pi}(\pi) \geq \theta_1)\\ &\text{and } (V_{\mathbb{R}}(R,\pi) \geq \theta_2)]\\ &\Longleftrightarrow (\exists x)\,[x \epsilon T_{pr}(R,\pi)\\ &\text{and } (x \neq \phi)], \qquad \phi = \text{empty set}\end{aligned}$$

This simply says that for $T_{pr}(R,\pi)$ to be defined, the value of the position, and the value of the rule with respect to the position, must be sufficiently high, i.e., greater than θ_1 and θ_2, respectively.

A position in the tree represents a subgoal of the parse, i.e., some channels have been identified and others remain to be identified. A *position* is defined as a 6-tuple, containing the following information:

(1) *Bit string* being operated on.

(2) *Set of found channels* {<*name, value, confidence measure, position in bit string, hypothesis used*>}.

(3) *Synchronization descriptor.*

(4) *Value of this position* $V_{\Pi}(\pi)$.

(5) *List of rules used to get to this position.*

(6) Information about non-local hypotheses that affect adjacent channels.

The *bit string* is arbitrary length bit string being operated on (usually 1 or 2 "words" long). The *set of found channels* is a list of the channels that have been identified at this position. The *confidence measure* of a found channel is derived from the statistics of the terminals mentioned previously. The *hypothesis used* is part of a rule. It is described below. The *synchronization descriptor* gives the current synchronization of the subcommutated decks. It is described in *Paragraph d.* The *position evaluation* $V_{\Pi}(\pi)$ is heuristic. The position evaluation function will utilize:

(1) Confidence measure on found channels.

(2) Complexity of rules (the more complex the rules, the less confident the measure).

(3) Depth of the tree (the deeper in the tree, the better the position).

(4) Distance of this channel from other previously identified channels.

A rule description is similar in concept to a position description; i.e., it should describe the parameter to be searched for, and any transformations that may be used on the input data stream. A rule consists of the following:

(1) *Name* of channel affected.

(2) *Width* of channel (number of bits).

(3) *Location*, a number (not a bit position) indicating where in the high-deck this channel is normally found.

(4) *Hypothesis set,* a set of all possible alternative hypotheses to be used to find this channel. This will be discussed shortly.

(5) *Value* of the rule.

The value of the rule is obtained from the *sampling rate* and the *complexity index*. The sampling rate is simply an indicator of how often a given commutator structure will cause this parameter to appear in words. This is, in some sense, how often this rule should have been used. The complexity index is associated with each hypothesis, and is inversely proportional to the complexity of the hypothesis.

An individual rule represents all possible ways of locating a specific channel. Each rule R_i has a hypothesis set.

$$H = \{h_1^{R_i}, h_2^{R_i}, \cdots, h_j^{R_i}\}$$

Each hypothesis is based on the errors that have been introduced by the communication channel on the bits in question, i.e., bits have been added, dropped, complemented, etc. The confidence measure of the hypothesis is determined by the statistics of the errors.

Before applying a hypothesis, we must first calculate a location in the bit stream at which the hypothesis is applied. This is done as follows:

(1) Calculate *distance* (number of bits) between channel being hypothesized about and the *anchor* or sync channel being used.

(2) Obtain the location in the bit string of the anchor.

(3) Obtain the location where the hypothesis is to be applied as the location of the anchor plus the distance.

d. Linguistic representation of the commutator structure. There are two main reasons for desiring to define the commutator structure linguistically. Firstly, it establishes the grammar rules that are to be used in the parsing process. Secondly, it supplies a suggested representation system for synchronization information. (The *synchronization descriptor* mentioned earlier.)

For specifying the commutator structure, we will use a phrase structure grammar. The following special symbols are used:

V_{NN} non-terminal non-subcommutated channel names = {A, B, C ···}

V_{NA} non-terminal subcommutated channel names = $\{\alpha\beta\gamma\delta\cdots\}$

V_N non-terminals = $V_{NN}\ U\ V_{NA}$

V_T terminals = {a,b,c, ···}

spatial concatenation

| temporal concatenation

Each deck of the commutator is represented by a grammar rule. Each subcommutated channel in a deck is represented by a non-terminal $\epsilon\ V_{NA}$. Each non-subcommutated channel is represented by a non-terminal $\epsilon\ V_{NN}$. Ad-hoc productions rewrite non-subcommutated non-terminals into terminals. A new production (not ad-hoc) is added for each subcommutated channel.

The following grammar rules describe the commutator shown in Fig. 27:

$$S \rightarrow A\,\alpha\,C\,D\,\beta\,E\,\delta\,F$$

$$\alpha \rightarrow G \mid \delta \mid H \mid I$$

$$\beta \rightarrow J \mid K \mid L \mid M$$

$$\gamma \rightarrow N \mid O$$

$$\delta \rightarrow P \mid Q$$

$$A \rightarrow a$$

.

.

$$Q \rightarrow q$$

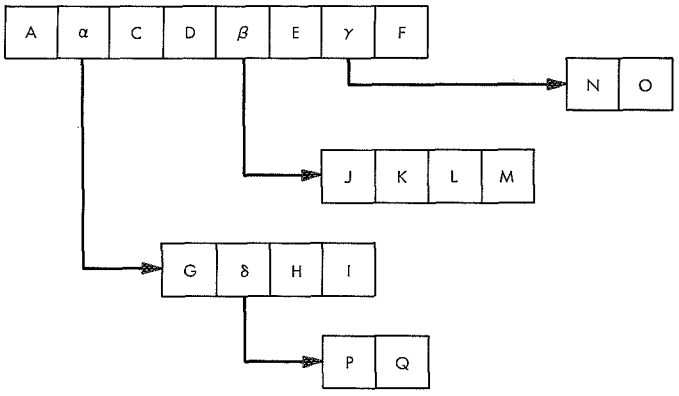

Fig. 27. A commutator structure

The interpretation of the linguistic representation of the commutator structure is straightforward. The linguistic representation also forms the basis for the representation of synchronization information. The synchronization descriptor could be given by a vector giving the position last recognized for each grammar rule with a member of V_{NA} on the left. In the example of Fig. 26, the vector

$$(3, 2, 1, 2)$$

would indicate that the last recognized channels in commutator positions α, β, γ, and δ were H, K, N, and Q, respectively. The next expected commutator words should then be

A I C D L E O F

A G C D M E N F

A P C D J E O F

etc.

Reference

1. Duda, R. and Hart, P., "Experiments in the Recognition of Handprinted Text," *AFIPS Proceedings Fall Joint Computer Conference*, pp. 1139–1150, 1968.

4. A Comparison of Methods for Synthesis of Correlated Noise, C. Travis

a. Introduction. This article compares two methods of generating a stationary time series with a prescribed spectral density function. Each method generates a sequence of numbers x_k which can be regarded as having been derived from a continuous time series $x(t)$ by sampling the values of the signal at spacing Δt, i.e., $x_k = x(t_k) = x(k\Delta t)$.

b. Theory.

Method I. In method I (Ref. 1), a sequence of independent normal random numbers with zero expectation and unit variance is passed through a linear filter to obtain the desired time series. Linear filtering is the process by which a set of input data y_k is transformed into a set of output data x_k by means of the relationship

$$x_k = \sum_{n=-L}^{L} c_n y_{n+k} \tag{1}$$

where the c_n are suitably chosen weights. The spectrum of the output of a linear filter, when the input is a stationary process, is

$$\Gamma_x(w) = R(w)^2 \Gamma_y(w) \tag{2}$$

where $\Gamma_x(w)$ and $\Gamma_y(w)$ represent the spectrum of the output and input, respectively (Ref. 2). The c_n in Eq. (1) and the $R(w)$ in Eq. (2) are related by the following formulas:

$$c_n = \frac{1}{w_0} \int_0^{w_0} R(w) \cos\left(\frac{n\pi w}{w_0}\right) dw \tag{3}$$

$$R(w) = \sum_{n=-\infty}^{\infty} c_n \cos\left(\frac{n\pi w}{w_0}\right) \tag{4}$$

where $R(w)$ is a symmetric periodic function with period $2w_0$, and the c_n can be considered as having been sampled from a continuous time series $c(t)$ at a spacing of $\Delta T = 1/2w_0$. It is only necessary to use a finite number of c_n in Eq. (4) to approximate $R(w)$ quite accurately. This corresponds to replacing $c(t)$ by

$$c^*(t) = \begin{cases} c(t) & \text{if } |t| \leq T^* \\ 0 & \text{if } |t| > T^* \end{cases}$$

The degree of accuracy obtained in approximating $R(w)$ depends only on where $c(t)$ is truncated. Once the desired degree of accuracy has been decided upon and a step size Δt has been chosen, the L in Eq. (1) can be determined by $L\,\Delta t = T^*$. Thus, for a given degree of accuracy, the smaller Δt, the more c_n are required.

Now, suppose it is desired to generate a sequence of numbers with spectral density function $F(w)$, $0 \leq w \leq w^*$, where this sequence is considered as having been derived from a continuous signal $x(T)$ by sampling at a spacing of Δt.

In Ref. 2, it is shown that white noise which has been sampled at a spacing of Δt has spectral density function

$$\Gamma_y(w) = \Delta t, \qquad -w_0 \leq w \leq w_0$$

where $w_0 = 1/2\Delta t$. The input data y_k can be considered as such a sampling. Using the above formula, Eq. (2) becomes

$$\Gamma_x(w) = \Delta t\, R(w)^2$$

Thus, if the c_n in Eq. (1) are obtained from Eq. (3) using

$$R(w) = \begin{cases} \left[\frac{1}{\Delta T} F(w)\right]^{1/2} & \text{if } 0 \leq |w| \leq w^* \\ 0 & \text{if } w^* < |w| \leq w_0 \end{cases} \quad (5)$$

the sequence x_k will have spectral density function

$$\Gamma_x(w) = \Delta t\, R(w)^2 = \begin{cases} F(w) & \text{if } 0 \leq |w| \leq w^* \\ 0 & \text{if } w^* < |w| \end{cases}$$

(Note that the requirement $w^* \leq w_0$ imposes a maximum on the step size Δt.)

Method II. The technique of method II is similar to that given in Ref. 3. Suppose $F(w), 0 \leq w \leq w^*$ is the prescribed spectral density function. This function will be approximated with spikes of power at frequencies chosen uniformly, but randomly, in the interval $[0,w^*]$. Let this interval be divided into N equal parts, and $w_k, k = 1, \cdots, N$, be chosen from a sequence of independent random numbers with a uniform distribution over the interval

$$\left[(k-1)\frac{w^*}{n}, k\frac{w^*}{n}\right], \qquad k = 1, \cdots, N.$$

The power in each of these intervals will be approximately $F(w_k)$ multiplied by the length of the interval, or $F(w_k)\, w^*/n$. Let $\phi_1, \cdots, \phi_2$ be a sequence of independent random numbers with a uniform distribution over $[0,2\pi]$. Then, if

$$A_k = \left[2\frac{w^*}{n} F(w_k)\right]^{1/2}$$

the time series

$$x(T) = \sum_{k=1}^{n} A_k \cos(2\pi w_k T + \phi_k) \quad (6)$$

will have power $F(w_k)\, w^*/n$ at the frequencies w_k and zero power elsewhere. As n is increased, the spectral density function of Eq. (6) will approach $F(w)$.

c. Programming techniques.

Method I. $F(w)$, Δt, and L (the number of c_n to be used) were inputs to the program. $F(w)$ was given in the form of a table of equally spaced values. $w_0 = 1/2\Delta t$ and $R(w)$ was calculated using Eq. (5). The c_n were obtained from Eq. (3) using the trapezoidal rule for numerical integration, and unknown values of $R(w)$ were found by linear interpolation. Then, a sequence (the sequence y_k) of independent normal random numbers was developed on the computer. This sequence was converted into the desired sequence by the sliding summation Eq. (1).

Method II. $F(w)$, Δt, and ND (defined below) were inputs to the program. $F(w)$ was given as a table of equally spaced values $F(f_J)$, $J = 1,2, \cdots, M$. ND was the number of random frequencies $w_k^{(J)}$, $k = 1, \cdots, ND$, to be chosen in each of the intervals $[f_J, f_{J+1}]$. The value of F at $w_k^{(J)}$ was found by linear interpolation.

To make what follows simpler, it will be assumed that only two values of F were given, its values at f_1 and f_2. Let $[f_1, f_2]$ be divided into ND equal parts, and $w_k, k = 1, \cdots, ND$, be a random frequency chosen in each of these subdivisions. The power in each subdivision is approximately

$$F(w_k)\frac{(f_2 - f_1)}{ND}$$

Let

$$A_k = \left[2F(w_k)\frac{(f_2 - f_1)}{ND}\right]^{1/2}$$

Then, by Eq. (6)

$$x_J = \sum_{k=1}^{n} A_k \cos(2\pi w_k J \Delta T + \phi_k)$$

is the sequence of numbers to be generated. The cosines on this equation were calculated using a recursion formula. A flow chart in Fortran notation for the process is shown in Fig. 28.

d. Comparison of methods.

Dependence on step size. Method I is very dependent on step size. As an example, consider the function

$$F(w) = \begin{cases} 0 \text{ if } 0 \leq w < 2 \\ 1 \text{ if } 2 \leq w \leq 3 \\ 0 \text{ if } 3 < w \end{cases} \quad (7)$$

Suppose it is desired to approximate this function to within 0.05 except at points of discontinuity where the error is allowed to exceed this value only in an interval

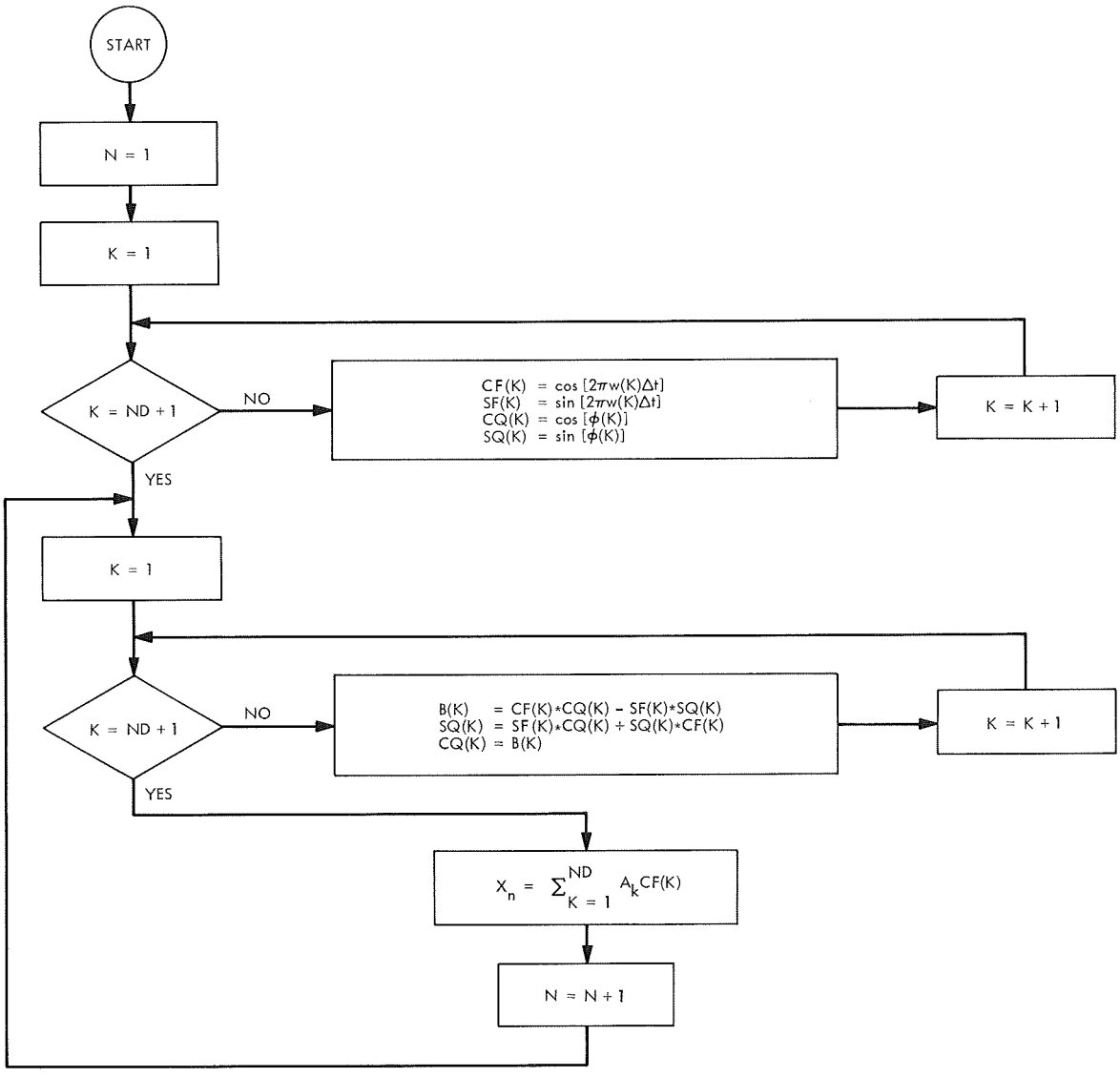

Fig. 28. Flow chart in Fortran notation for method II

of length 1/16 on either side of the points of discontinuity. The following table shows the relationship between Δt, w_0, and L (the number of c_n taken) when the above degree of accuracy is maintained.

Δt	w_0	L
0.166	3	15
0.04	12.5	60
0.01	50	240

Note this table is consistent with the relationship $L \Delta t = T^*$ which was stated earlier.

Method II is not dependent on step size.

Accuracy. The retrieval of the spectrum of a time series $x(t)$ is accomplished with two formulas:

$$0(T) = \frac{1}{T}\int_0^T x(t)\, x(t+T)\, dt \tag{8}$$

$$\int(w) = 2\int_0^S \phi(T)\cos(2\pi w T)\, dt \tag{9}$$

However, the choice of T and S in these two formulas is critical. T and S determine how long the time series $x(t)$ and the autocorrelation Function (8) are sampled. To understand the problems involved, one must realize that Eq. (9) does not give the "true spectrum" of $x(t)$, but rather a statistical estimation of it. As with most statistical estimators, in using Eq. (9) one is forced to compromise between variance and bias. Increasing S decreases the bias but increases the variance. Decreasing S produces the opposite effect. The extent to which fine detail can be detected in the spectrum of a time series is influenced by both the variance and the bias. Both should be made as small as possible. This can be accomplished by increasing the time T that the time series is sampled. Thus, it is possible to estimate the "true spectrum" to as much accuracy as desired if T is made large enough. However, for a fixed T, there must be a trade-off between variance and bias.

In discussing the accuracy of the two methods, the above considerations must be kept in mind. There are actually two types of accuracy involved. One is how close is the actual spectrum of the time series generated to the prescribed spectrum. The other is how close is the statistical estimation of the actual spectrum to the prescribed spectrum.

It is possible to determine what the actual spectrum of the two methods should be. As can be seen from Eq. (5), the prescribed spectrum of method I is $F(w) = \Delta t\, R(w)^2$ where $R(w)$ is defined by Eq. (4). The time series actually generated will have spectrum Δt at $R^*(w)^2$ where

$$R^*(w) = \sum_{N=-L}^{L} c_n \cos\left(N\frac{\pi w}{w_0}\right) \tag{10}$$

Thus, to determine how the actual spectrum of method I deviates from the prescribed one, Eq. (10) must be compared with Eq. (4).

In method II, the actual spectrum of the time series generated is a series of spikes at the random frequencies and zero power elsewhere. However, if one is not interested in resolving these spikes, and chooses T and S accordingly, the statistical estimation of the actual spectrum can come quite close to the prescribed spectrum. It was found that if 16 random frequencies were used in the interval (Refs. 2 and 3) that Function (7) could be reproduced to within an accuracy of 0.05.

Speed. Both methods were programmed in Fortran IV and run on a Univac 1108. Method I takes an average of $31*L + 38$ μs per generated point, where L is the number of c_n taken in Eq. (1). Method II takes an average of $31*ND + 5$ μs per generated point, where ND is the number of random frequencies used in Eq. (6). For a given degree of accuracy, the running time for method I will vary with step size, whereas method II will not. Demanding the same degree of accuracy for both methods, the following table shows how running time in μs varies with step size for spectral density Function (7).

Δt	Method I	Method II
0.166	503	501
0.04	1898	501
0.01	7478	501

Storage. Method I requires 256 locations for the subroutine with 61,228,902 locations for data when $L = 15$,

60, and 240. Method II requires 213 locations for the subroutine with 118 locations for data when $ND = 16$.

e. Conclusions. Method I compares favorably with method II only when using the maximum step size possible. In all other cases method II is superior.

References

1. Bykov, V. V., *A Method of Modeling Stationary Normal Noise on a Digital Computer.*
2. Jenkins, C. M., and Watts, D. C., *Spectral Analysis and Its Applications.* Holden-Day, Inc., San Francisco, Calif., 1968.
3. Peabody, P. R., and Adorno, D. S., "Digital Synthesis of Correlated Stationary Noise," *Communs. Assoc. Comput. Mach.*, Vol. 5, No. 7, Jul. 1962.

5. A Switch Controller for the SCU Cone, R. B. Kolbly

a. Introduction. This article describes a prototype microwave switch controller for use with the DSS 14 S-band cassegrain ultra (SCU) cone. This switch controller provides control of three microwave switches, prohibits selection of an improper transmitter load, and protects against microwave switch failure. Provision is made for remote operation, and a pictorial switch position display is provided.

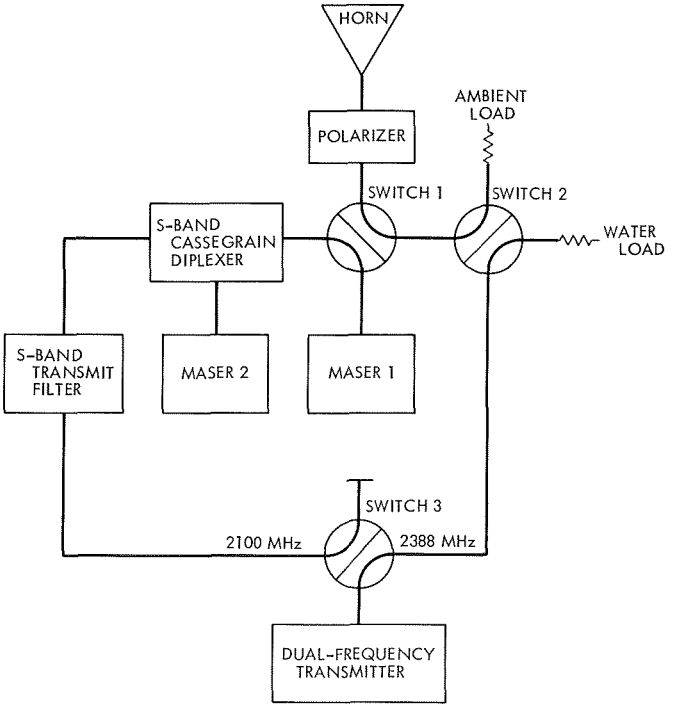

Fig. 29. SCU feedcone modification 2

b. Design. Figure 29 is a block diagram of the SCU feedcone and related microwave equipment. Inspection will show that there are certain switch combinations which could result in the transmitter being connected to a low-power termination or into a traveling-wave maser. To guard against any unfortunate incidents, the switch position information is applied to a relay-logic circuit which, through the external beam interlock circuit, will not allow the transmitter to operate unless all waveguide switches are in an acceptable configuration.

Individual control of each waveguide switch is available on the switch controller panel (Fig. 30). Also, any one of four operational modes (maser I, radar transmit, diplex and water load) may be selected either locally by push buttons on the front panel or remotely by contact closures. This allows remote control of a planetary radar mission.

Display of switch position is provided by panel lamps on the controller and also by projection-type displays in a block diagram of the microwave system on the front panel of the controller. These projections are visible in Fig. 30, along the upper edge of the panel.

c. Waveguide switch protection. Failure of a waveguide switch is usually caused by one of the control relays internal to the switch failing to disable the motor at the travel limit or a rotor "frozen" between positions. In either case, power remains applied to the stalled switch motor until either a fuse blows or the motor destroys itself.

To prevent these failures, a circuit is provided which senses the length of time power has been applied to the switch, and if it is excessive, power is removed and a fault is signaled to the operator.

Figure 31 is a schematic diagram of the switch protective circuitry. A small current transformer T1 senses the current to the waveguide switches. This 60-Hz ac voltage is rectified and clipped (so the output voltage is essentially independent of the number of switches drawing current) and used to charge a capacitor C1 through resistor R3. When capacitor C1 is charged to a set value, it operates a Schmitt trigger and relay K1. Relay K1 is in a latching circuit so the power remains off the waveguide switches until the current is manually reset. Provision is made for testing by applying a small dc input signal, which will check for satisfactory operation.

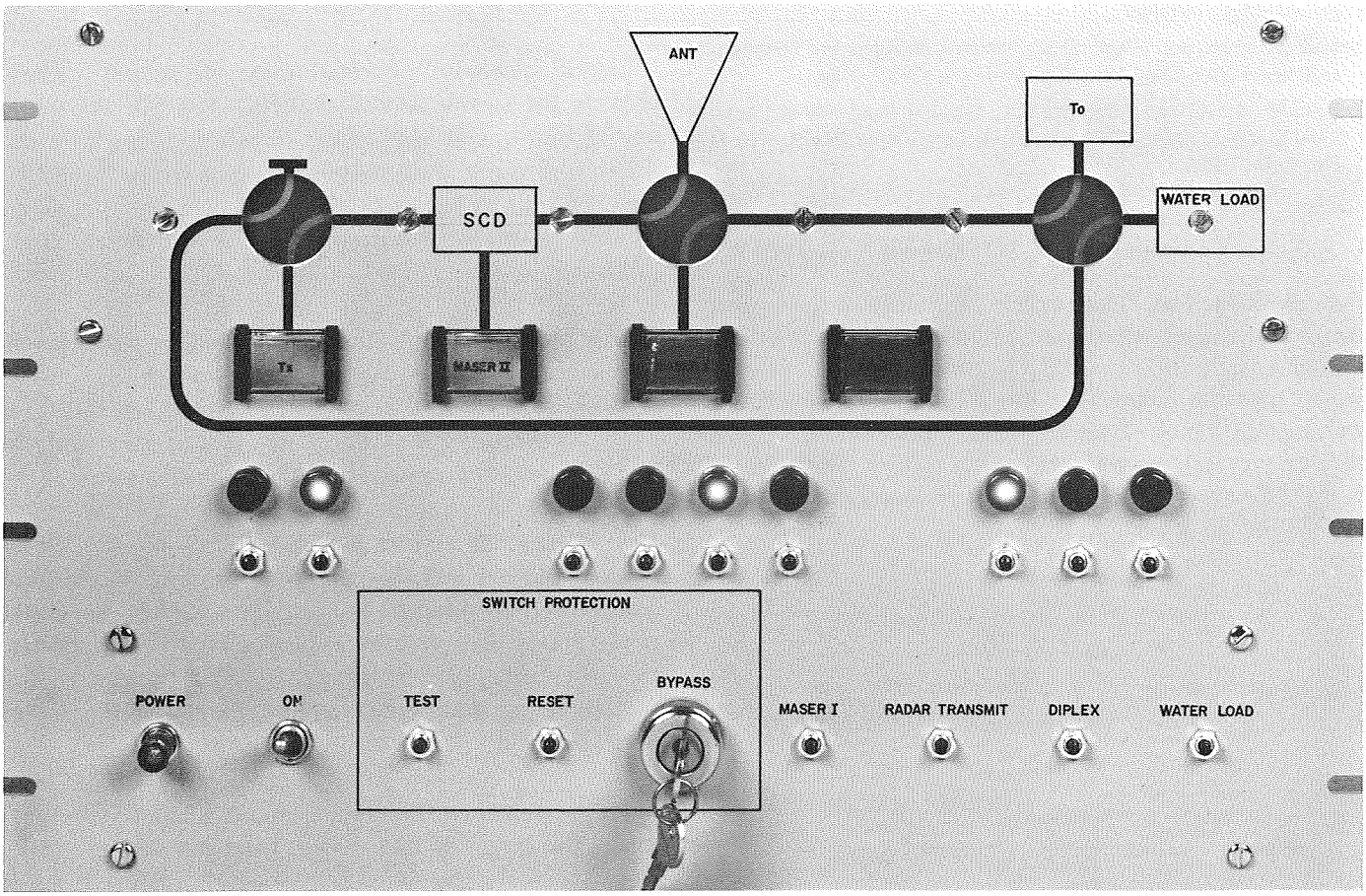

Fig. 30. Front view of SCU feedcone microwave switch controller

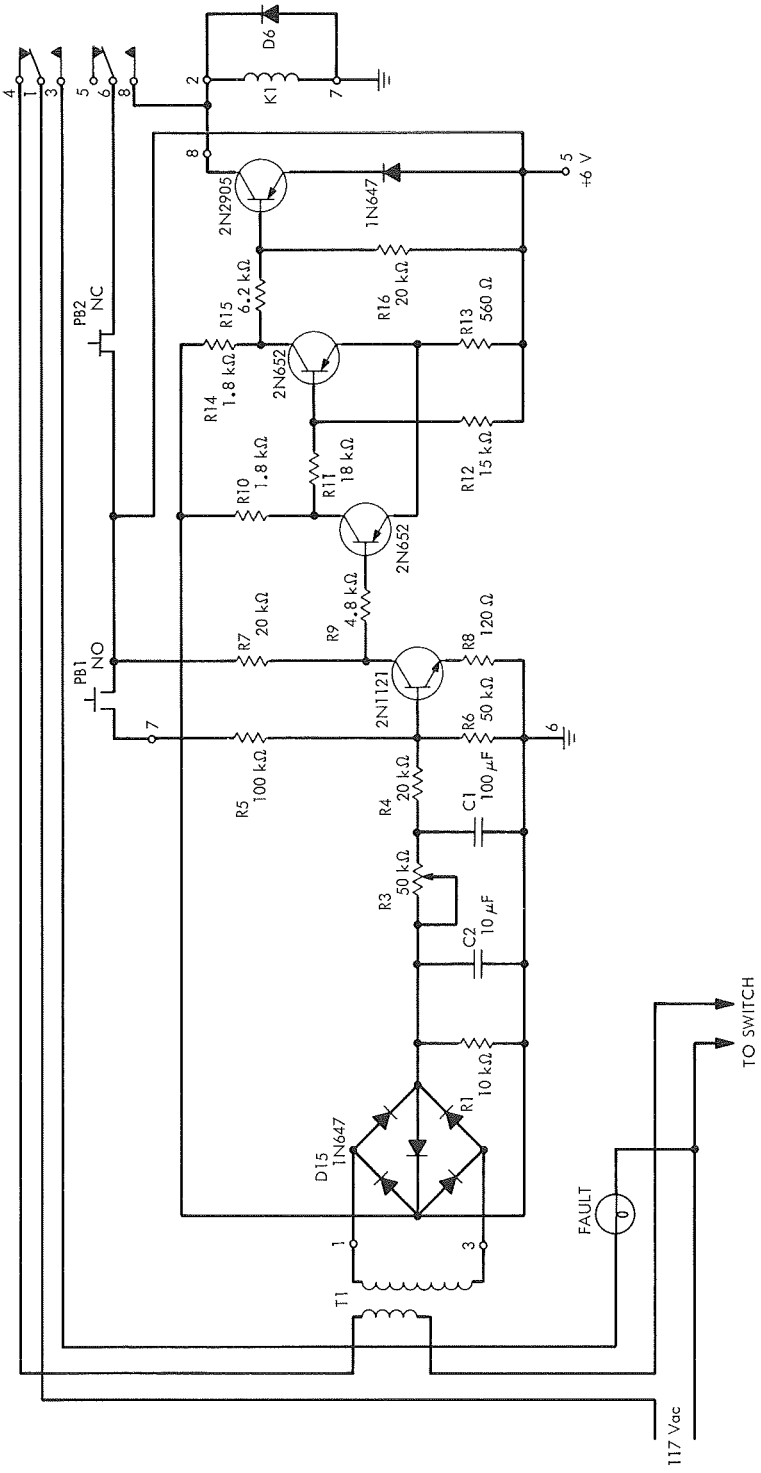

Fig. 31. Schematic diagram—microwave switch protection circuit

6. Automatic (Computer-Controlled) Ephemeris Update Tracking, *R. M. Gosline*

Automatic boresighting of antennas on coherent or noncoherent sources would be of benefit in DSIF operations, and would offer the possibility of automatic tracking of spacecraft, with only a small amount of signal degradation, without the use of monopulse feed systems. Previous testing at DSS 13 has indicated that the approach initially chosen was not optimum.

Considerable analysis of an ephemeris update tracking scheme has been made and will be summarized here, although experimental verification has not yet been attempted.

The method is to offset the ephemeris angles of a source with time-varying constant amplitude sinusoids that are shifted 90 deg in each axis. The antenna beam will, thus, trace a conical scan around the source, and if the sinusoids are correlated with a voltage proportional to the received noise (for radio sources) after suitable filtering, the boresight error may be determined and used to modify the ephemeris angles in near real-time.

The offset function is characterized by

$$X = r\cos wt, \qquad y = r\sin wt \tag{1}$$

The angular distance from the source to the center of the beam is

$$\phi(t) = [(r\cos wt - E_x)^2 + (r\sin wt - E_y)^2]^{1/2}$$

where E_x, E_y are the angular boresight errors. Solving for wt yields

$$wt = \sin^{-1}\left[\frac{r^2 + E_x^2 + E_y^2 - \phi^2(t)}{2r(E_x^2 + E_y^2)^{1/2}}\right] - \tan^{-1}\frac{E_x}{E_y} \tag{2}$$

The received noise is a function $N(\phi)$ of this distance and could be characterized by $|\sin A\phi(t)/A\phi(t)|$. With an appropriate digital filter, analysis is simplified and the system behavior made more predictable. Consider then

$$\left.\begin{aligned} N(\phi) &= 1, \qquad \phi(t) \le \phi \\ N(\phi) &= 0, \qquad \phi(t) > \phi \end{aligned}\right\} \tag{3}$$

where ϕ is the 3-dB point. The correlation coefficients are

$$C_x = \int_0^{2\pi/w} N(\phi)\cos wt\, dt, \qquad C_y = \int_0^{2\pi/w} N(\phi)\sin wt\, dt \tag{4}$$

and, using Eq. (3), become

$$C_x = \sin\theta_B - \sin\theta_S, \qquad C_y = \cos\theta_S - \cos\theta_B \tag{5}$$

where θ_B is the scan angle where the source enters the beam and θ_S is the scan angle when the source leaves the beam.

If $r \neq \phi$, a dead zone will exist at the center where the source will either be always in the beam ($r < \phi$) or always outside the beam ($r > \phi$), yielding no boresight information. It is reasonable then to choose $r = \phi$, and from Eq. (2)

$$\theta = \sin^{-1}\frac{(E_x^2 + E_y^2)^{1/2}}{2r} - \tan^{-1}\frac{E_x}{E_y}$$

so that

$$\left.\begin{aligned} \theta_S &= \sin^{-1}\frac{d}{2r} - \tan^{-1}\frac{E_x}{E_y} \\ \theta_B &= \pi - \sin^{-1}\frac{d}{2r} - \tan^{-1}\frac{E_x}{E_y} \end{aligned}\right\} \tag{6}$$

with

$$d = (E_x^2 + E_y^2)^{1/2}$$

Combining Eqs. (5) and (6), it can be shown that

$$C_x = E_x\left(\frac{4}{E_x^2 + E_y^2} - \frac{1}{r^2}\right)^{1/2}, \qquad C_y = E_y\left(\frac{4}{E_x^2 + E_y^2} - \frac{1}{r^2}\right)^{1/2} \tag{7}$$

and

$$E_x = FC_x, \qquad E_y = FC_y \tag{8}$$

with

$$F = r\left(\frac{4}{C_x^2 + C_y^2} - 1\right)^{1/2} \tag{9}$$

The sign of E_x and E_y may be resolved by assigning the sign of C_x and C_y, respectively. Thus, the boresight error may be found by equating Eqs. (8) and (9), using the correlation coefficients C_x and C_y.

For the unlikely case where $E_x = E_y = 0$ exactly, then $C_x = C_y = 0$ and special provisions must be provided because F becomes infinite. It may be advantageous to choose r slightly greater than ϕ to avoid this point of instability at the price of a small dead zone.

7. DSS 13 Operations, *E. B. Jackson and R. M. Gosline*

a. Experimental activities. From August 16 through October 15, 1969, DSS 13 conducted monostatic and bistatic planetary radar experiments with the planet Venus serving as a target. During bistatic experiments, the system was configured in a total spectrum, cross-polarized mode, while during monostatic experiments, the system was configured in a ranging and total spectrum, matched polarization mode. During this period, the range to Venus increased to 223.5×10^6 km and is still increasing.

Clock synchronization transmissions expanded to include DSSs 14, 41, 42, 51, and 62, with all stations reporting successful data reception, particularly during the period immediately succeeding the *Mariner* Mars 1969 encounter. A transient-induced failure in the programmed oscillator caused extensive lost time; however, full operation was restored on October 7, 1969, with a new transmitting coder being placed into service to correct some irregularities in correlated time at the receiving station.

In preparation for more extensive OSS antenna tests during later missions, the *Apollo 11* lunar surface experiments package was monitored for several days, with signals as strong as −119 dBm being measured.

The asteroid Geographos made its closest approach to the earth (9×10^6 km) on August 27, and an attempt was made to receive signals reflected from it after being transmitted, using the 450-kW R&D transmitter, from the 85-ft-diam antenna at DSS 13. However, due to a number of equipment failures, including the 450-kW klystron and elements of the water cooling system, insufficient integration time was obtained for signals to be detected. Geographos' next close approach will be in 1983.

During testing of a new spacecraft spectrum analysis computer program and associated hardware, observation of several radio sources, among them W3A, VYCMA, W49, and Orion, was made in an effort to detect an emission line for HDO, a form of heavy water where one hydrogen atom has been replaced by a deuterium atom. On-site data reduction did not disclose this line, and it is felt considerably longer observation times will be necessary if this line is to be observed at Venus.

Among the other effects predicted by the general relativity theorem is bending of electromagnetic waves during passage through a gravitational field. Although very slight, it should be possible to observe this predicted effect under appropriate conditions for waves which pass close by the sun. Utilizing DSSs 13 and 14 as a special long baseline interferometer, experimenters from California Institute of Technology carefully measured the position of quasars 3C273 and 3C279 for several days during the period when 3C279 was occulted by the sun. A position offset on the order of 1.25 arc secs is predicted for 3C279 by the general relativity theorem, although observation of this apparent offset may be masked by system instabilities and the effects of the sun's plasma field. Although some system problems were encountered, between 25 and 30 h of observing were performed over the period September 29 through October 14, with occultation of 3C279 taking place on October 8. Extensive data analysis will be necessary before system performance can be specified and conclusions drawn.

b. System performance.

Digital systems. A major system failure of the SDS 910, antenna positioning subsystem, programmed oscillator, and clock synchronization coders occurred as a result of a momentary short circuit of the 110-Vac line in the SDS 910, inadvertently caused by an operator performing routine maintenance. Forty-two integrated circuits were replaced in the interface logic between the above subsystems. Investigation is being conducted toward providing additional circuit protection. All other digital subsystems performed satisfactorily.

Antenna (electromagnetic). A new, wideband cassegrain feed cone with which the 85-ft-diam antenna has been equipped was described in SPS 37-59, Vol. II, pp. 93–95. Additional measurements of radio sources have been made to evaluate antenna efficiency with this new cone. Measurements of the apparent source temperature of Cygnus A indicate efficiencies of 61% at 2295 MHz and 57% at 2388 MHz. Data analysis indicates that mismatch between the feedhorn and the subreflector may be contributing to the low efficiency at 2388 MHz, but further tests are necessary to isolate the difficulty.

Transmitter (S-band). The 450-kW klystron sprang a leak in the drift tube coolant circuit. The leak was caused by a crack in the klystron coolant fitting, which was located in such a position that local repair was not practical. The tube was returned to the manufacturer for repair and the spare klystron, which had been previously repaired by the manufacturer, was installed. Unfortunately, it also failed (vacuum leak) after a short period of service, leaving an inoperative transmitter with which to conduct the Geographos experiment. Although expedited return and installation of the first klystron was performed, insufficient time remained during which Geographos was close enough for the experiment to be attempted. The already short operating time was further reduced by a failure of a pump which circulates the transmitter coolant, and the experiment was unsuccessful.

c. *System improvements.*

Digital Systems. Software capability of the station control and monitor (SCAM) subsystem was expanded with the addition of a SCAM program tape editing program written for the SDS 930. Means are provided to automatically shorten a SCAM program tape by removal of corrected corrections, inserting only the latest corrections and punching out only non-zero words. With this capability, the fast and slow coder programs were combined (under breakpoint selection) with SCAM internal checkout routines on a program tape one-half the length of either of the original coder programs.

The radiometer program used at DSS 14 with gas-tube noise sources was modified to run on the DSS 13 SDS 930 with solid-state noise diodes. The SDS 930 has been interfaced with binary-coded decimal time, and binary angles from the angle readout subsystem have been connected for use in the servo automation program.

8. Hi-Rel Module Development, *D. W. Slaughter*

a. Low-level interface module. A low-level interface module has been designed in the format of the family of Hi-Rel digital modules (SPS 37-46, Vol. III, pp. 159–161), and provides four identical interface circuits. Each circuit has two inputs. One input is compatible with most integrated circuits. The other input detects signal zero-crossings and provides an interface with certain DSIF RF systems which present a binary signal with levels of +2 and −2 V across 50 Ω. The circuit output has the same characteristics as the Hi-Rel *nand* gate.

Figure 32 is a circuit diagram. Input 1 has a logic threshold in the range of 1.2 to 2.1 V (worst-case), which

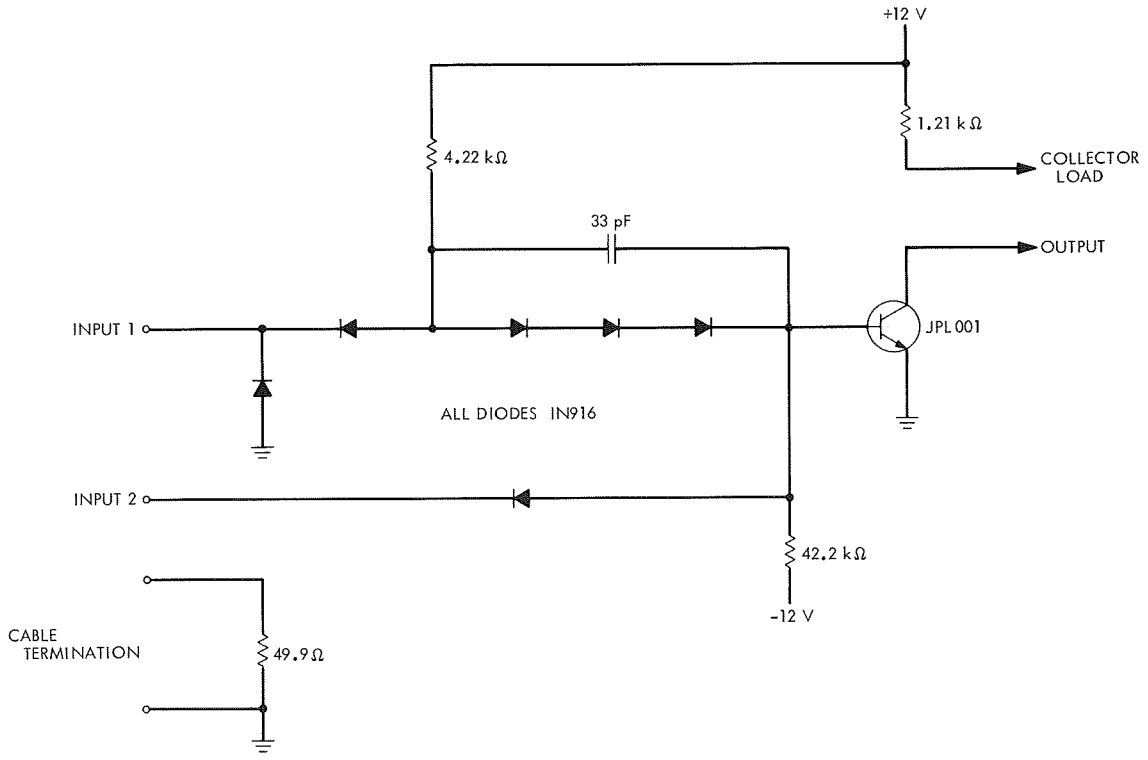

Fig. 32. Low-level interface circuit

is compatible with diode–transistor logic and transistor–transistor logic integrated circuits using 5-V power supplies. The input circuit is diode back-clamped to prevent ringing on the interface wiring, which could otherwise result in spurious output responses. Input 2 has a logic threshold in the range of 0 to 0.2 V. A 50-Ω resistor is available for terminating 50-Ω coaxial cables. The circuit output will be the *nand* function of the two inputs, if both are used.

b. Integrated circuit applications. As reported in SPS 37-57, Vol. II, pp. 121–123, a project is underway which will qualify integrated circuit (IC) devices for use in DSIF mission-independent digital equipment.

The DSIF Hi-Rel digital modules program (SPS 37-46, Vol. III, and later issues) provided the DSIF with a standardized approach to the construction of digital equipment. The benefits of this program have been listed as an input to our IC program.

(1) Standardization permits much of the time-consuming portion of the total manufacturing effort to be performed prior to the final completion of the logic design. Basic logic circuits are pre-wired into a module whose size is identical for all circuits. Connections between modules are made by taper-pin wiring. Assembly hardware is also standardized and prototype equipment can be rapidly assembled after the logic design is complete.

(2) The use of the same module types on several different equipments in various stages of design and construction allows for the modest stocking of these modules. Original estimates of module quantity or mix may be adjusted between projects.

(3) Costs are reduced. The number of DSIF stations (requiring copies of a given equipment) is modest; however, there are a broad spectrum of system functions to be mechanized. Thus, considerable cost savings accrue when it is possible to use the same logic devices to implement various system functions.

(4) Compatible logic modules are available for future activity, including equipment modification, additional spares, and new station equipment.

The benefits of standardization will accrue for ICs only if the chosen family is continuously available. JPL ownership of the design is not in itself sufficient to ensure availability. If the basic approach does not have widespread usage and quickly becomes obsolete, production facilities will be shut down. Maintenance of these facilities under sustaining contract from JPL could be costly. Therefore, an attempt is being made to identify a family of ICs which has achieved wide-spread acceptance, is manufactured by several competent companies, and is likely to experience graceful obsolescence. It appears possible to make a choice from one of the currently available transistor–transistor logic families which meet the above criteria. The remaining effort consists of:

(1) Selecting and qualifying preferred circuits.

(2) Deciding on the degree of control which JPL must exercise over the manufacturing process to ensure reliability.

(3) Developing procurement and test specifications.

(4) Selecting and certifying vendors from those available and interested.

The following is a discussion of the packaging problems being evaluated: Connections to the Hi-Rel logic modules were made through a taper-pin connector block integral to each module. The additional series connections of a plug and socket were not necessary since the equipment repair philosophy called for the replacement of functional assemblies (which contain a number of modules and do have a quick disconnect plug) at the individual stations. Replacement of modules by pulling taper pins is readily accomplished at the class A repair depot. Unfortunately, the taper-pin block, if used as the connector for IC standard logic modules, would not provide adequate logic density for many of our future equipments. The required logic density is a function of limitations on the length of the wires connecting the logic elements. These limitations derive from the speed capability of the chosen logic family, noise thresholds which are lower than those of our discrete component circuits, and the system logic complexity. A search is underway for a logic packaging scheme which will yield the required density while retaining the benefits of standardization which have accrued from our existing Hi-Rel modules.

The IC packages themselves may represent the smallest number of logic module types available in a form which can be pretested and stocked. Unit logic modules carrying several IC packages (unit logic brings all logic element inputs and outputs to the module connector pins) may be an archaic carry-over from a package form used with discrete components. Furthermore, the unit logic module packaging format tends to be incompatible with

medium-scale integrated (MSI) and large-scale integrated (LSI) circuits which are a form of functional (special-purpose) packaging.

The following general outlines describe a packaging scheme which may meet our objectives. Plugable functional subassemblies carry a significant number of IC packages. These subassemblies accept IC packages on standard grid dimensions. The standard grid dimensioning permits the blank subassembly to be premanufactured, and after mounting the IC packages, to be quickly intraconnected by a tape-controlled automatic machine. Special-purpose modules carrying discrete components, hybrid microelectronics, or LSI will fit the standard grid pattern being constrained to some multiple of the standard dimensions. The key to the acceptability of the whole packaging scheme just described is the technology and methodology for attaching the IC packages.

D. Tracking and Navigational Accuracy Analysis

1. Introduction, *T. W. Hamilton and D. W. Trask*

The DSN Inherent Accuracy Project was formally established by the DSN Executive Committee in July 1965. The objectives of the project are:

(1) Determination (and verification) of the inherent accuracy of the DSN as a radio navigation instrument for lunar and planetary missions.

(2) Formulation of designs and plans for refining this accuracy to its practical limits.

Achievement of these goals is the joint responsibility of the Telecommunications and Mission Analysis Divisions of JPL. To this end, regular monthly meetings are held to coordinate and initiate relevant activities. The project leader and his assistant (from the Mission Analysis and Telecommunications Divisions, respectively) report to the DSN Executive Committee, and are authorized to task project members to (1) conduct analyses of proposed experiments, (2) prepare reports on current work, and (3) write descriptions of proposed experiments. The project is further authorized to deal directly with those flight projects using the DSN regarding data-gathering procedures that bear on inherent accuracy.

The various data types and tracking modes provided by the DSIF in support of lunar and planetary missions are discussed in SPS 37-39, Vol. III, pp. 6–8. Technical work directly related to the Inherent Accuracy Project is presented in SPS 37-38, Vol. III, and in subsequent *Deep Space Network* SPS volumes, and is continued in the following subsections of this volume.

The first five articles (*Subsections 2–6*) of this section report on work carried out under the Tracking System Analytical Calibration (TSAC) activity. The TSAC activity provides calibration of tracking data and estimates of DSN parameters whose uncertainties represent limitations to navigational accuracy. In addition to the generation of analytic calibration coefficients, TSAC validates their proper transmission/utilization during a mission and performs detailed post-flight analysis of the DSN tracking data to uncover/resolve any anomalies which may exist. Further, this activity is concerned with defining the present inherent limitations to navigational accuracy and recommending feasible improvements which will reduce these limitations to meet future navigational accuracy requirements. This information is presented for the consideration of the DSN and flight projects in negotiating error levels for these parameters consistent with the navigational requirements of the projects and economical constraints.

The major guideline for TSAC in the formulation of an error budget is to maintain a balanced system of error sources. That is, the system is balanced such that a given expenditure of resources is budgeted to minimize the rss of the resultant navigational errors. In general, this necessitates advancing the state-of-the-art for the most critical error sources and, within the above constraints, reducing the effects of other error sources to a level which is negligible when compared to the most critical error source.

For a mission such as *Mariner* Mars 1969, the tightest bounds on the allowable errors for a number of parameters arise from the navigational accuracy requirements during encounter support. In particular, encounter navigational accuracy is most sensitive to error sources, which cause a diurnal signature on the radio tracking data (see Hamilton and Melbourne, SPS 37-39, Vol. III, pp. 18–23). These sources of error are of two classes: (1) those parameters which define the location of the DSS in inertial space, and (2) those phenomena which directly affect the DSS tracking data. The first category includes the location of the DSS with respect to the earth's crust, Universal Time (UT1), polar motion (the motion of the earth's crust with respect to the spin axis), precession and nutation (orientation of the earth's spin axis with respect to inertial space), and the ephemerides of the earth, moon, and target body. Of these, uncertainties in the first three are currently the major limitations to encounter support navigational accuracy. Utilizing the information on UT1 supplied by the